Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

Claim 1 (currently amended):

A program flow method in a program component system, comprising a running time system and several components each having one program portion, said method comprising the following steps during the execution of the program portion of a first component:

a) data acquisition by means of the running time system, of data of a second component into said first component independent of ~~program-defined~~ programmer-defined interfaces in said second component; and

b) data disposal by means of the running time system, of data of said first component into said second component independent of ~~program-defined~~ programmer-defined interfaces in the second component.

Claim 2 (previously amended):

The method according to claim 1, characterized in that the data transmitted during the data acquisition are transferred from a memory image portion of said second component into a transfer data region of said first component, and/or that the data transmitted during the data disposal are transferred from a transfer data region of said first component into a memory image portion of said second component.

Claim 3 (previously amended):

The method according to claim 1, characterized in that said data acquisition and/or data disposal is carried out without the cooperation of said second component.

Claim 4 (previously amended):

The method according to claim 1, characterized in that said second component is inactive during said data acquisition and/or data disposal.

Claim 5 (previously amended):

The method according to claim 1, characterized in that said transfer data region of said second component is located in a saving region during said data acquisition and said data disposal.

Claim 6 (previously amended):

The method according to claim 1, characterized in that local and/or non-persistent data of said second component are transmitted during said data acquisition and/or said data disposal.

Claim 7 (previously amended):

The method according to claim 1, characterized in that during the execution of the program portion of said first component a waiting list is made indicating which of the data of said first component require data disposal.

Claim 8 (previously amended):

The method according to claim 1, characterized in that a called component an directly access an access data region comprising the data fields defined and /or available in said calling component.

Claim 9 (previously amended):

The method according to claim 1, characterized in that the call of a component is triggered by call information comprised in a docking point of the calling component.

Claim 10 (currently amended):

A method of expanding a program component system comprising several components, by one further component, said method comprising the steps of:

a) searching for docking points for said further component in said program component system, which docking points correspond to an inheritance parameter determined by a definition of said further component; and

b) modifying the components of the program component system where at least one docking point was found, by entering call information about the further component at each docking point found; wherein said expansion of said program component system is completed without any expansion interface of said several components being defined by a programmer.

Claim 11 (previously amended):

The method according to claim 10, characterized in that all interaction interfaces of the previous components are predefined as potential docking points.

Claim 12 (previously amended):

The method according to claim 10, characterized in that all interaction screen fields referenced by the previous components and/or all print mask output fields and/or all access operations on persistent data are predefined as potential docking points.

Claim 13 (previously amended):

The method according to claim 10, characterized in that by entering said call information into a docking point a call of the further component from the component into which said call information was entered, is prepared.

Claim 14 (previously amended):

The method according to claim 10, characterized by an additional step of:

c) generating at least one binary object from the definition of the further component.

Claim 15 (original):

The method according to claim 14, characterized in that a maximum of one binary object is generated for each docking point that has been found.

Claim 16 (previously amended):

The method according to claim 15, characterized in that while generating each binary object, the memory allocation is considered in the one component of the program component system which includes the underlying docking point.